

データマイニングと情報可視化

Week 2

稲垣 紫緒

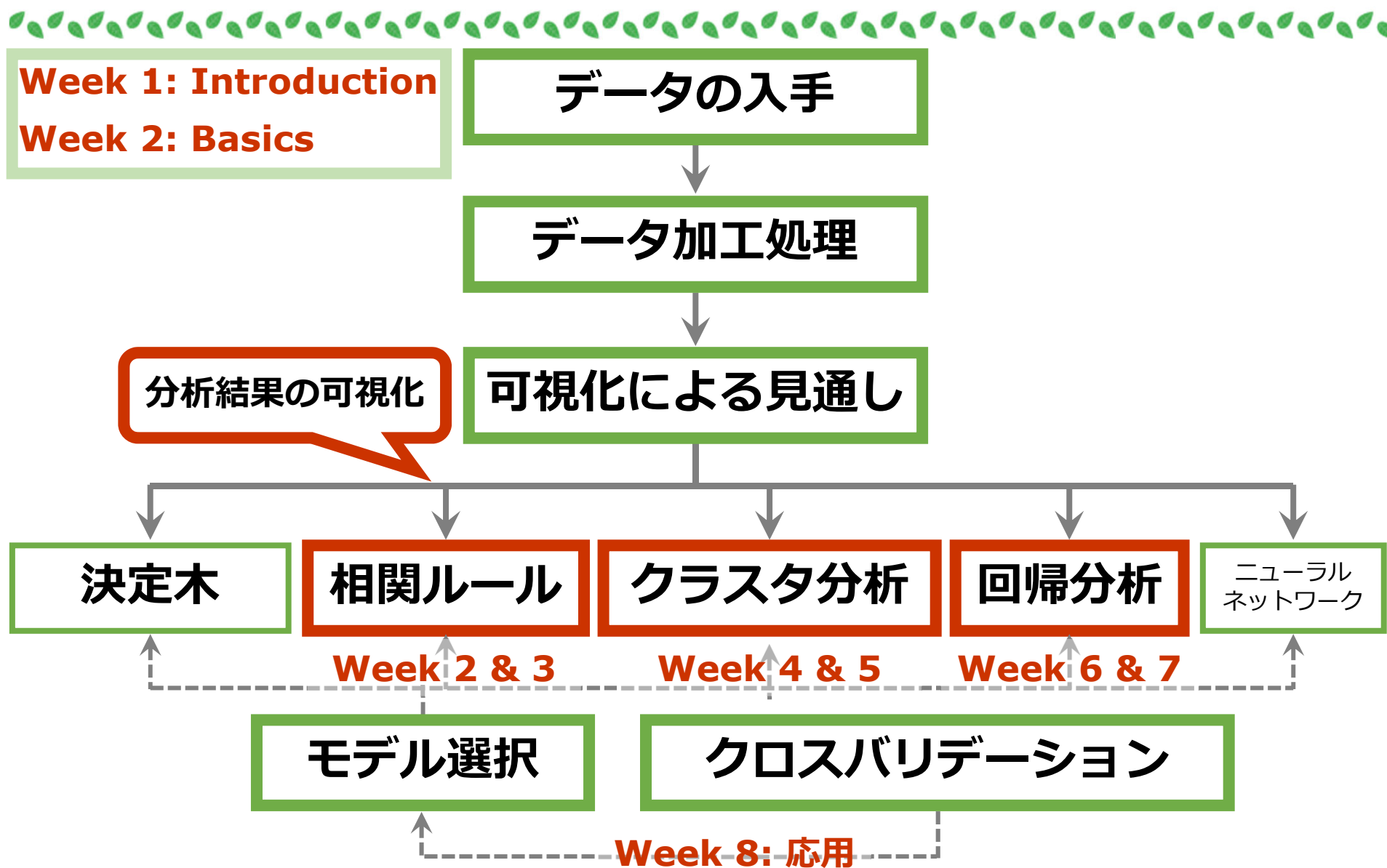
いながき しお

理学研究院 物理学部門 / 共創学部

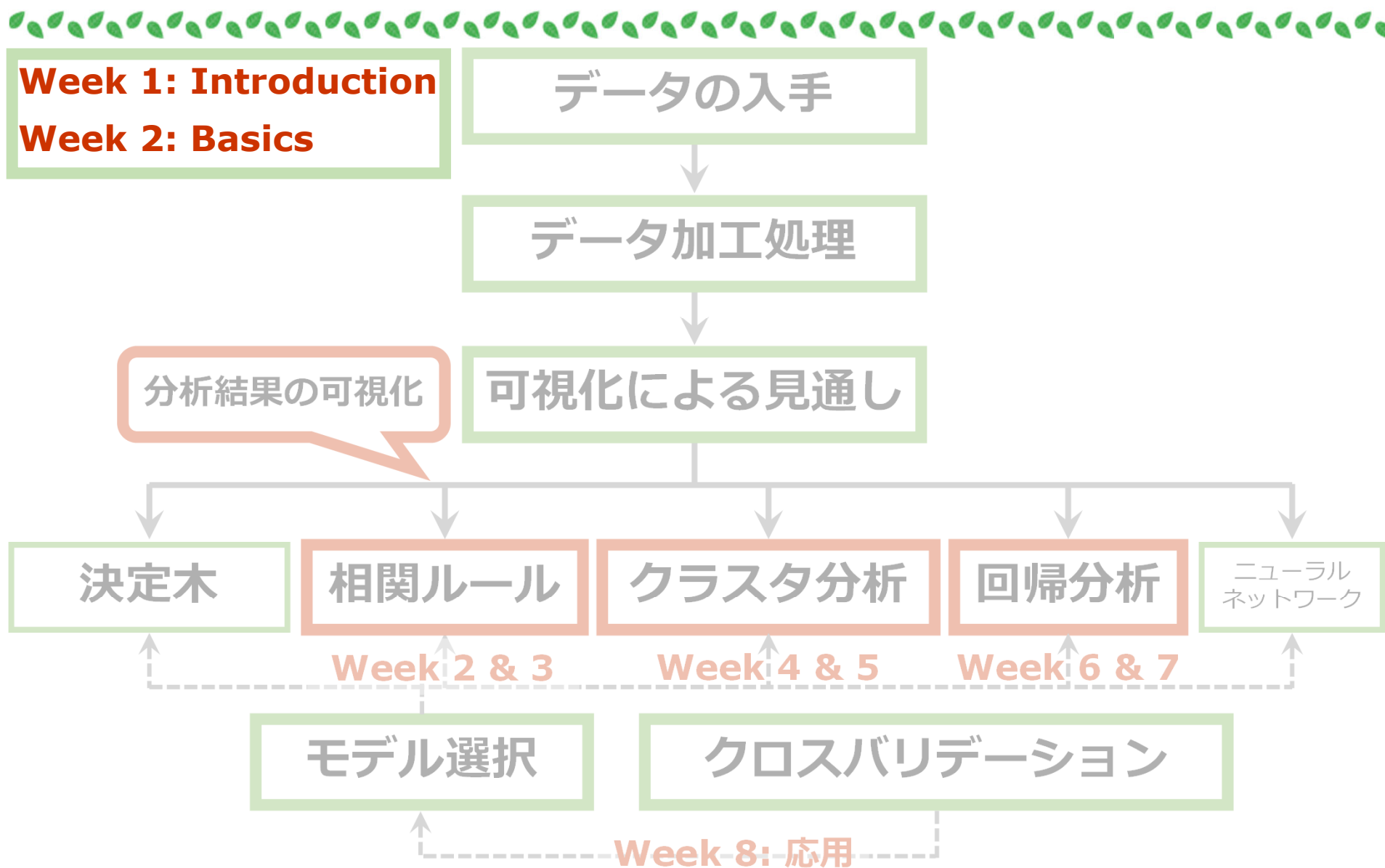
inagaki@phys.kyushu-u.ac.jp

ウェスト1号館 W1-A823号室

授業計画



授業計画



Pythonのライブラリ



行列演算

→ Numpy, SciPy

描画・可視化

→ matplotlib, seaborn,
NetworkX

データフレーム処理

→ Pandas

機械学習

→ scikit-learn

トピックモデル

→ gensim

深層学習

→ PyTorch, TensorFlow



Pythonの基礎



集合変数 / Aggregate variable



- リスト / List

score=[48, 52, 33, 88, 67, 85, 91, 75]

x=[**5** **2** **6** **8** **3** **5** **8** **9** **5** **3**]

- 辞書型 / Dictionary

dic_data = {'apple': 100, 'banana': 100}
apple -> 100

- 集合型 / Set

配列 / Array



■ 数 / Scalar

4

■ ベクトル / Vector

5 2 6 8 3 5 8 9 5 3

■ テンソル / Tensor

1	3	8	4	6	2	7	9	0	3
2	6	2	7	9	0	2	4	6	7
5	1	4	3	7	5	8	4	9	9

Data type @ Pandas



■ Series: 1次元データ

Seriesオブジェクトを作るには、Seriesを使います。

```
ser = pd.Series([10, 20, 30, 40])
```

■ DataFrame: 二次元データ

DataFrameオブジェクトを作るには、DataFrameを使います。

```
df = pd.DataFrame([[10, "a", True],  
                  [20, "b", False],  
                  [30, "c", False],  
                  [40, "d", True]])
```


type



配列にはいろいろな種類があります。

リストの時もあるし、DataFrameのときもあるので、確認するためにはtypeという関数を使います。

Series

1次元データです。Seriesオブジェクトを作るには、Seriesを使います。

Series is a one dimensional array. Use Series to creat a Series object.

```
[9]: 1 import pandas as pd
      2 ser = pd.Series([90, 100, 80, 70])
      3 ser
      4 type(ser)
```

```
[9]: pandas.core.series.Series
```

PandasのSeries



type



DataFrame

2次元データです。DataFrameオブジェクトを作るには、DataFrameを使います。
DataFrame is a two dimensional array. Use DataFrame to creat a DataFrame object.

```
[12]: 1 suits = pd.DataFrame([[90, "a", 88],  
2                           [100, "b", 95],  
3                           [80, "a", 85],  
4                           [70, "c", 80]])  
5 print(suits)  
6 type(suits)
```

	0	1	2
0	90	a	88
1	100	b	95
2	80	a	85
3	70	c	80

```
[12]: pandas.core.frame.DataFrame
```

PandasのDataFrame

type

```
[10]: 1 # for文を使って、一行ずつ読み込むコードに書き換えてみよう
      2 # Rewrite the code to read data line by line. Use For structure.
      3 f = open("data/w2_sample1.txt", "r", encoding="utf-8")
      4 line = f.readline()
      5 while line:
      6     print(line)
      7     line = f.readline()
      8 #     lines = lines.append(line)
      9 f.close()
      10
      11 type(line)
```

データマイニングと情報可視化の授業です。

ファイル入出力の練習をしています。

頑張りましょう!!

[10]: str

STR→文字列

解析でエラーが出る時、入力する変数のタイプが正しくないためにうまくいかないことがよくあります。変数の種類(型)に気を付けましょう。

行名を指定: .index

```
df.index=["Harvey", "Mike", "Louis", "Harold"]
```


	0	1	2
Harvey	90	a	88
Mike	100	b	95
Louis	80	a	85
Harold	70	c	80

列名を指定: .columns



```
df.columns=["Math", "Phys", "Biology"]
```

	Math	Phys	Biology
Harvey	90	a	88
Mike	100	b	95
Louis	80	a	85
Harold	70	c	80



.head / .tail



`df.head()`

最初の5行を表示

`df.head(10)`: 数字を指定すると、その行数分表示

`df.tail()`

最後の5行を表示

`df.tail(10)`: 数字を指定すると、その行数分表示

shape / size / len



* 行数を調べる : `len(df)`

* 列数を調べる : `len(df.columns)`

* 行数と列数を調べる : `df.shape`

* 要素数を調べる : `df.size`

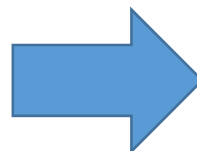
列を抽出

```
suits_Math = suits.Math
```



```
suits_Math = suits['Math']
```

	Math	Phys	Biology
Harvey	90	a	88
Mike	100	b	95
Louis	80	a	85
Harold	70	c	80



Harvey	90
Mike	100
Louis	80
Harold	70

複数列を抽出



1列を抽出 / Extract one column.

```
suits_Math = suits['Math']
```

複数列を抽出 / Extract multiple columns.

```
suits[['Math', 'Phys']]
```

かっこが二つ必要です。

平均値 / Average



数学の平均値を出す

```
suits_av = suits.Math.mean()
```

```
suits_av = suits['Math'].mean()
```

```
suits_av = suits.mean()
```

DataFrameに直接.mean()をかけると、
それぞれの列の平均値を出す。

数字出ないデータも入っていることがあるので、その場合は

```
suits_av = suits.mean(numeric_only=True)
```

数字の場合だけ計算する、って言うオプション付けるの推奨

標準偏差/ Standard deviation



数学の標準偏差を出す

```
suits_av = suits.Math.std()
```

```
suits_av = suits['Math'].std()
```

```
suits_av = suits.std()
```

DataFrameに直接.std()をかけると、
それぞれの列の標準偏差を出す。


CSVファイルの読み込み:.read_csv



```
X= pd.read_csv('data/w2_student-mat.csv')
```

セパレータがカンマの時はオプションは不要。

```
class,sex,weight,height,time  
A,F,45,150,85  
A,M,50,160,80  
A,F,55,155,74  
B,M,78,180,90  
B,F,51,158,65  
B,M,40,155,68  
C,F,80,185,90  
C,M,86,175,81  
C,F,52,162,73
```



CSVファイルの読み込み:.read_csv




```
X= pd.read_csv('data/w2_student-mat.csv', sep=';)
```

セパレータがカンマ以外の場合はオプションで指定する。

You can specify a delimiter with "sep=';'" as an option of import command, read_csv.

```
class;sex;weight;height;time  
A;F;45;150;85  
A;M;50;160;80  
A;F;55;155;74  
B;M;78;180;90  
B;F;51;158;65  
B;M;40;155;68  
C;F;80;185;90  
C;M;86;175;81  
C;F;52;162;73
```



.groupby



```
student_data_math_group =  
student_data_math.groupby('sex')
```

➡ データを性別ごとに分ける

```
type(student_data_math_group)
```

➡ pandas.core.groupby.generic.DataFrameGroupBy

グループをDataFrameとして抽出★



Extract a group as a DataFrame

```
student_data_math_group = student_data_math.groupby('sex')
```

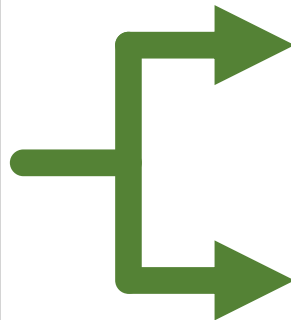


データを性別ごとに分ける/ Separate data by gender

年ごとに分けたり、クラスごとに分けたり、伝票番号ごとにわけたり...

class	sex	weight	height	time
A	F	45	150	85
A	M	50	160	80
A	F	55	155	74
B	M	78	180	90
B	F	51	158	65
B	M	40	155	68
C	F	80	185	90
C	M	86	175	81
C	F	52	162	73

.groupby('sex')



student_data_math_group

class	sex	weight	height	time
A	F	45	150	85
A	F	55	155	74
B	F	51	158	65
C	F	80	185	90
C	F	52	162	73

class	sex	weight	height	time
A	M	50	160	80
B	M	78	180	90
B	M	40	155	68
C	M	86	175	81

グループをDataFrameとして抽出

Extract a group as a DataFrame

```
Math_F = student_data_math_group.get_group('F')
```

➡ 女性(F)のグループを抽出 / Extract female data

student_data_math_group

class	sex	weight	height	time
A	F	45	150	85
A	F	55	155	74
B	F	51	158	65
C	F	80	185	90
C	F	52	162	73

class	sex	weight	height	time
A	M	50	160	80
B	M	78	180	90
B	M	40	155	68
C	M	86	175	81

Math_Fとして抽出

class	sex	weight	height	time
A	F	45	150	85
A	F	55	155	74
B	F	51	158	65
C	F	80	185	90
C	F	52	162	73

グループごとの平均値



```
student_data_math_group = student_data_math.groupby('sex')
```

```
student_data_math_group.mean()
```

➡ groupbyに直接.mean()をかけると、
それぞれのグループごとに、全ての列に対して平均値を出す

```
student_data_math_group.get_group('F').mean()
```

➡ 女性(F)のグループを抽出してから全ての列に対して平均をとる方法
Extract female data, then calculate the mean of all of the columns

```
student_data_math_group.get_group('F') ['G3'].mean()
```

➡ 女性(F)のグループを抽出してから、列 G3 の平均をとる方法
Extract female data, then calculate the mean of the column, G3.

列や行の削除: .drop

famrel の列を削除する。を削除するときは、オプションaxis=1を付ける。

```
df2= df.drop('famrel', axis=1)
```

もともとのDataFrameに削除した結果を反映させるには、オプション inplace=True を付ける。

```
df.drop('famrel', axis=1, inplace=True)
```



デフォルトはaxis=0で行を削除する。
行名「3」を削除したいときは

```
df2 = df.drop(3, axis=0)
```

削除した結果をdf2という別名のDataFrameに格納する。
削除した結果をもとのDataFrameに反映させたいときは、
オプション inplace=True を付ける。

条件を指定して抽出: .isin()

Extract columns or rows under a certain condition

シンプルに書くなら

```
score[score.Sex==1]
```

応用が利くのは

```
score[score['Sex'].isin([1])]
```

isin()は()の中にリストを入れたりisin(['F','M'])とできたりするので複数条件で抽出するときに便利です。

```
suits2 = suits[suits['Phys'].isin(['a'])]
```



数字の時にはクォーテーションマークはいりませんが、文字列の時にはクォーテーションマークが必要です。

複数条件を指定して抽出

Extract columns or rows under multiple conditions.

男性のデータを抽出

```
student_data_math [student_data_math['sex']=='M']
```

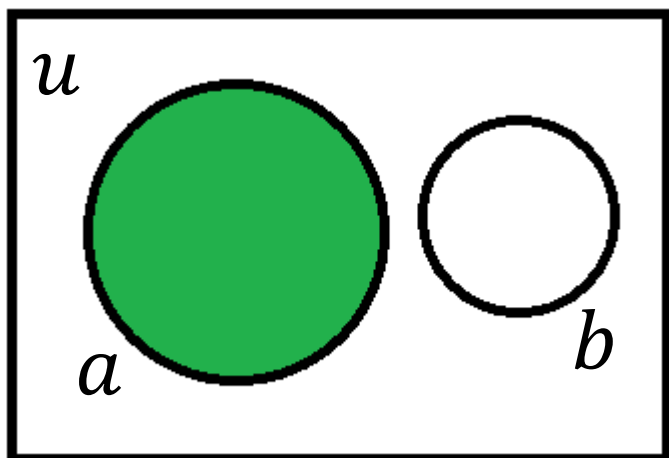
G3スコアが80点以上のデータを抽出

```
student_data_math[student_data_math.G3>=80]
```

男性のデータで、かつG3スコアが80点以上のデータを抽出

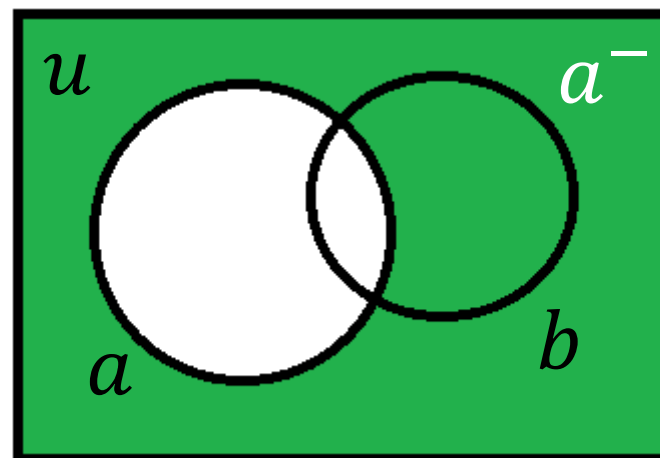
```
student_data_math[(student_data_math['sex']=='M') &  
(student_data_math.G3>=80) ]
```

集合 / Set



集合 a / Set a

$$a \in u$$

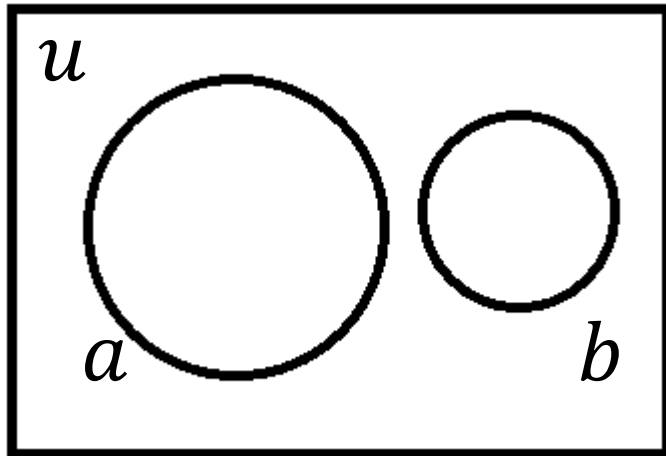


a の補集合 a'

a' the complement of a

for **Market basket analysis**

集合 / Set

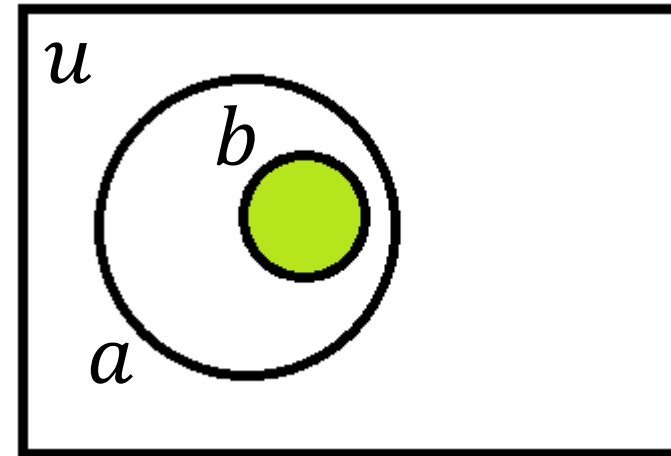


a と b は素集合

→ 共通の要素を持たない

a and b are disjoint sets

→ they have no element in common



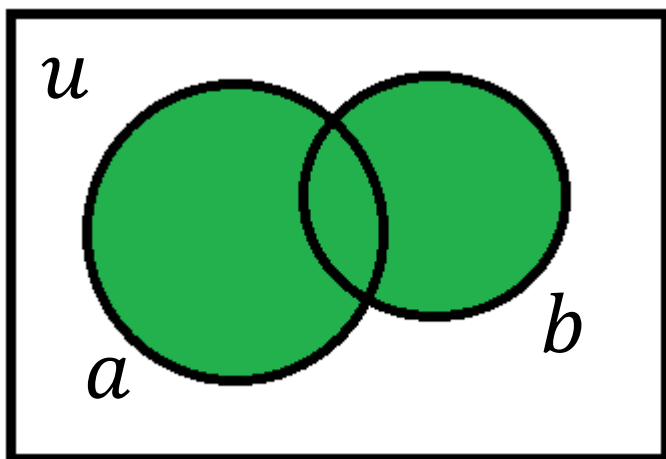
b は a の部分集合

b is proper subset of a

$$b \subset a$$

集合演算 / Set Operation

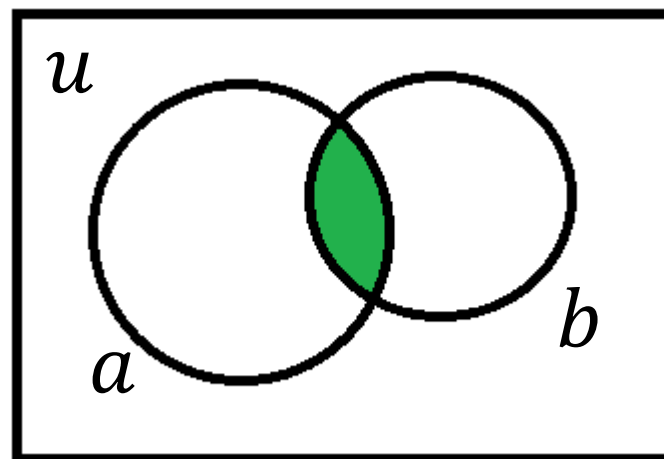
和 / Union



`a.union(b)`

$$a \cup b$$

積 / Intersection



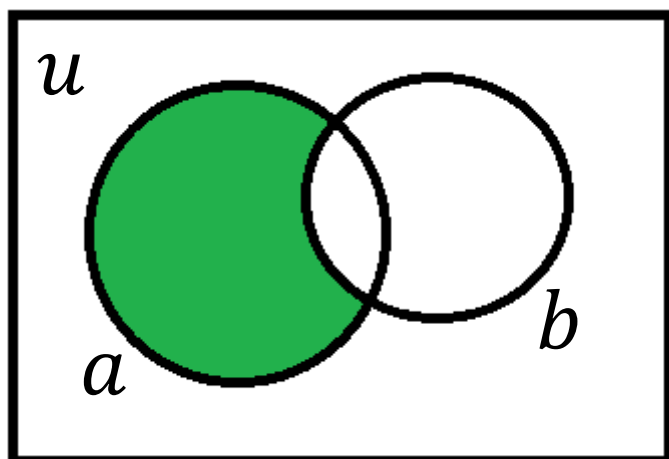
`a.intersect(b)`

$$a \cap b$$

for **Market basket analysis**

集合演算 / Set Operation

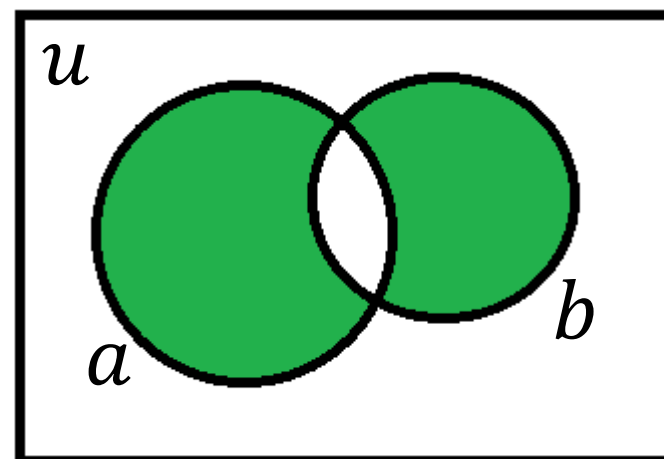
差/ Difference



`a.difference(b)`

对称差

Symmetric intersection



`a.symmetric_difference(b)`

for **Market basket analysis**

集合@Python



set型も同じく複数の値を格納できる型ですが、リスト型とは以下のような違いがあります。

- * **重複した要素がない**
- * **要素に順番がない**

集合を扱うための型です。

集合型は、集合を扱うので、**同じ要素が重複することができません。**

集合@Python



■ リスト型[1,2,3,3,2,1]を集合に変換

```
myset = set([1,2,3,3,2,1])
```

```
print(myset)
```

→ **{1, 2, 3}**

■ myset をリスト型に変換

```
mylist = list(myset)
```

```
print(mylist)
```

→ **[1, 2, 3]**

かっこの形に注意!!

その他集合の演算などは
演習問題で確認しよう!!